Amendments to the Specification:

Please amend the paragraph beginning on page 6, paragraph [0023] of the application to read as follows:

[0023] In accordance with the present invention, a real-time or near real-time distributed data mining and analysis system 11 is provided for use in open architecture systems. With reference to Fig. 1, a network software device, such as media server plug-in (server) 21 ~~in~~, for example, a content distribution system generally comprises a server program 23 (e.g., a web server or a media server) that serves data via a network and generates a log file 25 for storage in a local database. As ~~the server 21~~ a server, such as media serving system 14 of Fig. 2, serves information to a client, the log file 25 increases. An access module 27 accesses the local database and retrieves preferably only the newly added portion of the log file 25 (e.g., the information added since the last retrieval operation). The retrieved information, that is, a log string is transmitted to the network to a selected analyzer module, such as 29a. If the access module 27 uses, for example, Transmission Control Protocol (TCP), then the log string can be unicast to the analyzer 29. Alternatively, the log string can be unicast or broadcast to the analyzer module 29a if using User Datagram Protocol (UDP).

Please amend the paragraph beginning on page 6, paragraph [0024] of the application to read as follows:

[0024] The analyzer modules, such as 29a, represent software for implementing a state machine for storing and retrieving values for variables. They can be installed in a hierarchical manner to allow information from lower modules or programs 29a to be sent to upper modules 29b to merge the data. Thus, the analyzer modules 29, such as 29a, 29b, and 29b1, constitute a distributed, multi-layer analyzing tool which can process log data, for example, in a distributed and hierarchical manner so that the data transfer needed for reporting is significantly reduced to achieve essentially real-time reporting. Real-time reporting is particularly useful for streaming media. Since the analyzer ~~module 29 is~~ modules are designed to work in a distributed fashion, it is highly scalable. The analyzer modules ~~29~~ preferably analyze sequences of numbers and strings generated from software that understands analyzer module commands such as a parser module described below. Good uses are, for example, collecting real-time voting information, analyzing and aggregating real-time number sequence generated by media servers, or other specific applications.

Please amend the paragraph beginning on page 6, paragraph [0025] of the application to read as follows:

[0025] Basically, the analyzer module 29a ~~has two different modes. The~~ operates in a first mode (i.e., 'Model'), which is used to collect and analyze raw source data. As illustrated in Fig. 1, a number of network devices 21 provide source data to respective analyzer modules 29a operating in mode 1. The analyzer modules 29a each store analyzed data in memory in database form (e.g., table, records, and fields). Each analyzer module 29a is operable to manage multiple tables wherein each table may have multiple records and each record may consist of multiple fields. The main differences between a standard database and an analyzer module 29a database are that each record in an analyzer module 29a table can have different fields and each field can have multiple properties or multiple strings.

Please amend the paragraph beginning on page 7, paragraph [0026] of the application to read as follows:

[0026] As indicated in Fig.1, analyzer modules 29a can be configured to have parent-child relationships whereby one or more Model analyzer modules 29a are child modules instructed to report to a specified parent analyzer module 29b executing in the second mode (i.e., 'Mode2'). Similarly, a number of Mode2 analyzer modules 29b can be configured as child modules instructed to report to a specified parent Mode2 analyzer module 29b. Thus, Mode2 analyzer modules 29b can collect data from multiple Model analyzer module 29a instances and aggregate data from each connected child. Mode2 analyzer modules 29b can also connect to upper analyzer modules 29b1 also operating in mode 2 to push data.

Please amend the paragraph beginning on page 10, paragraph [0035] through page 11 of the application to read as follows:

[0035] The regional data centers 16 are located at strategic points around the Internet backbone. With reference to Fig. 4, a regional data center 16 comprises a satellite and/or terrestrial signal transceiver, indicated at 61 and 63, to receive inputs and to output content to users 20 or control/feedback signals for transmission to the NOC or another hierarchical component in the system 10 via wireline or wireless communication network. A regional data center 16 preferably has more hardware than a media serving system 14 such as gigabit routers and load-balancing switches 66 and 68, along with high-capacity servers (e.g., plural media serving systems 14) and a storage device 62. The CPU 60 and host 64 are operable to facilitate storage and delivery of less frequently accessed on-demand content using the servers 14 and switches 66 and 68. The regional data centers 16 also deliver content if a standalone media serving system 14 is not available to a particular user 20. The director agent software preferably continuously monitors the status of the standalone media serving systems 14 and reroutes users 20 to the nearest regional data center 16 if the nearest media serving system 14 fails, reaches its fulfillment capacity or drops packets. Users 20 are typically assigned to the regional data center 14 that corresponds with the Internet backbone provider that serves their ISP, thereby maximizing performance of the second tier of the distribution network 12. The regional data centers ~~14~~ 16 also serve any users 20 whose ISP does not have an edge server.

Please amend the paragraph beginning on page 11, paragraph [0038] through page 12 of the application to read as follows:

[0038] With reference to Fig. 5, the data flow of the distributed data mining and analysis system 11 of the present invention will now be described in the context of the content distribution system 10 for illustrative purposes. Fig. 5 depicts a real-time log-reporting application of the analyzer modules 29a. A data generating device in the data mining and analysis system 11 can be a media server (e.g., a plug-in in the media serving system 14 in Fig. 2). A parser module 41 and a Java XBM App server 43 are provided, respectively, as an input and final data processing application. The analyzer modules 29a, 29b, and 29b1 are used as dynamic log analyzing and aggregating tools and are deployed at one of the tiered devices 14, 16 and 18 or in the acquisition network 22, of Fig. 2, in the content distribution system 10, of Fig. 2.

Please amend the paragraph beginning on page 12, paragraph [0039] of the application to read as follows:

[0039] The parser module 41 is a tool that receives a log line generated by a media server plug-in 21 and parses its fields and field values. The access module ~~23~~ 27 operates in conjunction with the media server plug-in 21 to provide packets to the parser module 41 when events occur such as the beginning or end of a stream. When the access module sends a log line to the parser module 41, it adds information into the header to assist the parser module 41 with the identification of the type media server generating the log line. The parser module 41 has its own XML-based log definition file that describes which portion of log should be used as ~~a~~ an analyzer module field and how to create a table and record of the analyzer module 29a. The parser module 41 then sends a command to an analyzer module 29a to register a new variable and also sets a field value to each field. The parser module 41 is preferably the driver of the entire network 11 for creating and updating tables.

Please amend the paragraph beginning on page 12, paragraph [0040] of the application to read as follows:

[0040] The analyzer modules 29a, 29b, and 29b1 are generic statistics-analyzing tools. An analyzer module 29a gets commands from the parser module 41 and analyzes each field of a command based on the analyzing method of each field. Once the specified interval has elapsed, tables created in an analyzer module executing in Model are transmitted to the root tier analyzer modules 29b and 29b1.

Please amend the paragraph beginning on page 12, paragraph [0041] of the application to read as follows:

[0041] The root tier of analyzer modules 29b and 29b1 ~~pushes~~ push tables into the Java App server 43 using an XBM function call. The tables are then sent to be stored in a database 45 (e.g., an Oracle database) by the Java App server 43.

Please amend the paragraph beginning on page 12, paragraph [0042] through page 13 of the application to read as follows:

[0042] As stated previously, the media server plug-in 21 generates source information and sends it to the parser module 41 (e.g., using UDP). The parser module 41 parses each log line sent from different media server plug-ins 21 (e.g., ~~WMT~~ WMS server 52, Real G2 server 48, and the like) and generates commands using a configuration file for each media server type. The parser module 41 preferably uses an XML-based log definition file for processing each line. The XML-based log definition file describes how a log file 25, of Fig. 1, is organized, which field is to be processed, and how the field is to be processed. The parser module 41 determines which variables are to be stored in the analyzer module 29a and sets the variables with appropriate values by sending commands to the analyzer module 29a. The communication between the media server plug-ins 21 and the parser module 41, and between the parser module 41 and the analyzer module 29b is preferably UDP.

Please amend the paragraph beginning on page 13, paragraph [0044] of the application to read as follows:

[0044] Table 1: Real-Time Monitored Data

MOD		
	Current	Peak
~~WMT~~ WMS	564	654
Real	215	300
Total	779	954
On-Air		

	Current	Peak
~~WMT~~ WMS	564	654
Real	115	200
Total	679	854
On-Stage		
	Current	Peak
~~WMT~~ WMS	564	654
Real	215	300
Total	779	954

Please amend the paragraph beginning on page 14, paragraph [0046] of the application to read as follows:

[0046] The current connection number and peak values for each product and format combination are stored for the sampling duration of 5 minutes, for example. The lowest layer analyzer modules 29a therefore monitor the connection numbers for 5 minutes and send the sampled data to upper layer analyzer modules 29b. These analyzer modules 29b, in turn, collect information from the lower layer analyzer modules 29a and send the merged data to higher level analyzer modules 29b1.

Please amend the paragraph beginning on page 14, paragraph [0047] of the application to read as follows:

[0047] In order for the parser module 41 to divide the concurrent stream into different product-format types and send the right commands to the analyzer module 29a, the parser module preferably extracts the following parameters whenever it receives a log packet:

- account____(content provider name such as CNN, ABC etc.)
- product____(OnDemand, OnStage, OnAir)
- format[ ](media type such as Netshow, Real)
- asset[ ](media file name including the)

- starttime___(starting time of the stream)
- endtime[ ](ending time of the stream)

Table2: Sample URLs in the log packets

	Sample URL in the log
Dmd-ns	mms: / /10.0.3.40/cnn1/.asf
Air-ns	mms:/ /10.0.3.40/onair/cnn/2.asf
Stg-ns	mms:/ /10.0.3.40/v2/onstage/cnn/3.asf
Dmd-g2	cnn/dir1/1.asf
Air-g2	Ibeam/v2/onair/cnn/2.asf
Stg-g2	Ibeam/v2/onstage/cnn/3.asf
Dmd-qt	Rtsp:/ /10.0.3.40/cnn/1.asf
Air-g2	rtsp:/ /10.0.3.40/v2/onair/cnn/2.asf
Stg-g2	rtsp:/ /10.0.3.40/v2/onstage/cnn/3.asf

Please amend the paragraph beginning on page 15, paragraph [0049] of the application to read as follows:

[0049] When the parser module 41 receives a log packet, it extracts appropriate parameters from the packet (e.g., account, product, format, starttime, endtime and asset). If the packet is from a content provider that parser module has not processed before, it registers the required variables to the analyzer module 29a. For example, these variables can be presented in product-format form and defined in the <RegVarList> section in the configuration file. Whenever a stream is started, the parser module 41 sends a command to increase an appropriate field for the given content provider. When a stream is stopped, the parser module 41 sends a command to decrease the field by one for the content provider.

Please amend the paragraph beginning on page 16, paragraph [0051] of the application to read as follows:

[0051] In the GlobalSetting section, the local Internet Protocol (IP) address and port are used by the parser module to listen for the log packets that are sent by the log packet generator programs such as the media server plug-ins 21. Destination IP address and port are the address of an analyzer module 29a to which the parser module will send the data. Whenever the parser module sends a command to the analyzer module, it determines when the content provider was last registered to the analyzer module. If it passed more than RegisterInterval seconds, it will re-register the content provider to analyzer module.

Please amend the paragraph beginning on page 18, paragraph [0057] of the application to read as follows:

[0057] In the instructions list, many instruction sets can be defined. When a log is to be parsed, the instruction set is considered from the first one until the matching one is found. For each instruction set, it can have three kinds of attributes: NotContain, Contain, GeneratorId. ~~They~~ These attributes can be used by themselves or in combination. The NotContain attribute indicates that, if the log does not contain the specified substring, the instruction set is used. The Contain attribute indicates that if the log contains the specified substring, the instruction set is used. The GeneratorId attribute indicates that if the generator id is matched, then the instruction set is used.

Please amend the paragraph beginning on page 18, paragraph [0058] of the application to read as follows:

[0058] The analyzer module 29a can handle Number and String data types. In case of Number, the analyzer module processes a 'Null-Terminated' string as a string type representation of an integer. Therefore, it will be converted to 'int' type using 'atoi()' function. In the cease of String, the analyzer module regards handed 'Null-terminated' strings as C language's standard 'Null-Terminated' string representing some variable. The analyzer module keeps monitoring for data sent from other applications. It could be a sequence of numbers (e.g., 10, 15, 21,...) or a sequence of strings (e.g., Tomato, Apple, Orange, Apple,.....) related to each field type.

Please amend the paragraph beginning on page 18, paragraph [0059] through page 19 of the application to read as follows:

[0059] For Number type data, handed strings are converted into C language type 'int" to allow essentially any arithmetic operation to be performed with them. An analyzer module 29a has the ability to get several values from these number sequences, as shown in Table 3.

Table 3: Values for Number Sequences

Method	**Meaning**
Average	Average of total number sequence
Biggest Number	Biggest number out of entire sequence of numbers
Smallest Number	Smallest number out of entire sequence of numbers
Total	Total sum of ~~who~~ sequence of numbers
Average of Total	Average to total values
Biggest Total Number	Biggest number out of sequenced total value
Smallest Total Number	Smallest number out of sequenced total number

Please amend the paragraph beginning on page 19, paragraph [0060] of the application to read as follows:

[0060] A number analyzing example is shown in Table 4:

Table 4: Number Analyzing Sample

#Seq	Number Sent	Average	Biggest	Smallest	Total	Total Average	Total Biggest	Total Smallest
1	10	10	10	10	10	10	10	10
2	20	15	20	10	30	20	30	10
3	10	13.33	20	10	40	26.66	40	10
4	5	11.24	20	5	45	31.24	45	10
5	22	~~13/39~~ 13.40	22	5	67	38.39	67	10
6	32	~~16.49~~ 16.50	32	5	99	48.49	99	10

Please amend the paragraph beginning on page 21, paragraph [0067] of the application to read as follows:

[0067] From Sequence #1 to #3, to the analyzer module point of view, a new string appears. When the new string is sent, the analyzer module 29a allocates enough memory to store that string and keep track of hit counts for each string. Once a string is added, whenever the same string is received, the analyzer module simply adds to the hit count and recalculates the statistics.

Please amend the paragraph beginning on page 21, paragraph [0070] of the application to read as follows:

[0070] Fig. 1 above shows that multiple Mode 1 instances can be connected to a Mode 2 instance, and that a Mode 2 instance can send aggregated data to an upper level Mode2 instance. The analyzer ~~module 29~~ modules 29a, 29b, and 29b1 uses formulas to aggregate field types. Assuming each analyzer module mode1 instance in Fig. 1 has one number type and one string type variable, and each sends its information to analyzer module mode2, an analyzer module in Mode 2 collects data from different analyzer module Mode1 instances. How the analyzer module Mode2 aggregates multiple fields with data types Number and String will now be described.

Please amend the paragraph beginning on page 21, paragraph [0072] and continuing through page 22 of the application to read as follows:

[0072] The method of addition for each field's method property is not always the same. For example, in the case of 'Average', a total hit count for each average value is needed in order to add them. Assuming a two-field instance, A and B, and the hit count for each record is hA, hB, the average for each field is aA, aB. The formula to get the average is shown below:

$$\textit{Weighted Average} = \frac{\underline{(\cancel{HA}\ hA\ x\ aA + (hB\ x\ aB)}}{\cancel{hA - hB}\ \underline{hA + hB}}$$

Please amend the paragraph beginning on page 22, paragraph [0074] of the application to read as follows:

[0074] Table 7 above shows how an analyzer module applies number field aggregating rules. When pushed data arrives from an analyzer module in Model (1), an analyzer module in Mode 2 copies all fields into its database. After receiving data from connection (2), it adds those fields with the fields from (1). The row corresponding to Hit Count 15 of Table 7 is a good example to test the aggregating formula. Average value '7' is a result of following formula:

$$Weighted\ Average = \frac{\underline{(hA\ x\ aA) + (hB\ x\ aB)}}{hA\ [=]\ \underline{+}\ \cancel{H}\ \underline{h}B} = \frac{\underline{(5\ x\ 8) + (10\ x\ 6.5)}}{5 + 10} = 7$$

But 'total average" 54, is obtained from adding 22 with 32, not from averaging 22 and 32. In conclusion, no matter how many Model analyzer modules are connected to the analyzer module in Mode 2, field size never changes, because fields sent from the Model analyzer modules are compressed into a single field.

Please amend the paragraph beginning on page 23, paragraph [0075] of the application to read as follows:

[0075] For String type data, the same method is used to aggregate multiple fields. If a new string appears, that string is added and the statistics recalculated for each string.

Table 8: String Field Aggregating Simulation

Push Seq#	String Field Sent to analyzer module
1	Tomato: 50%(2), Banana: 50%(2)
Result	Tomato: 50%(2), Banana: 50%(2)
2	Tomato: 27.27%(3), Banana 27.27%(3), Lemon 45.45%(5)
Result	Tomato: 50%(5), Banana: 50%(5), Lemon: 33%(5)
3	Lemon: 40%(10), Apple: 40%(10), Pineapple: 20%(5)
Result	Tomato: 12.5%(5) Banana: 12.5%(5) lemon: 37.5%(15) Apple: 25%(10) Pineapple: 12.5%(5)

After receiving #1 instance, the analyzer module 29a copies it into its memory. When it receives #2 instance, it adds to the hit count, if the string is the same. If there is a new string, it adds that string and copies its hit count. Regarding the second result: 'Tomato' and 'Banana' were already in analyzer module Mode2'_memory, so it just adds the hit count (5=2=3). 'Lemon' was not, however, so 'Lemon' is added and the hit count set to '5'.

Please amend the paragraph beginning on page 23, paragraph [0077] of the application to read as follows:

[0077] Note in Table 9 below that the structure of each record in a table may be different, and that every record has its own name to distinguish it from others. In database management, "Name of Record' has a equal meaning to 'Primary Key' in a table. 'Apple', 'Banana' and 'Mango' in a 'Fruits' table is used as a primary key. If the string fields are considered, one field has a multiple string value in it. This is a significant difference between the string field in a typical database system and that of analyzer module.

Table 9: Example Fields in a Table

Table Name	Records	Fields	Fields Value
Fruits	Apple	Count (Num)	25
		Color (String	"Red":40%(10), "Green":60%(15)
		Weight (Num)	208
	Banana	Length (Num)	230
		Count (Num)	12
	Mango	Count (Num)	20
		Origin (String)	"Mexico":55%(11), "Hawaii":45%(9)

Cars	Porsche	Count (Num)	8
		Model (String)	“911”:12.5%(1), “928: 87.5%(7)
	BMW	Count (Num)	12
		Model (String)	“325I”:33%,(4), “525” :33%(4), “740I”: 33%(4)

Please amend the paragraph beginning on page 26, paragraph [0081] through page 27 of the application to read as follows:

[0081] The ‘SetField’ command is used to set a field value. Whenever a field value is set, related information, such as average, biggest, total, etc., are recalculated based on the new field value. If the specified table name or record with ‘Record ID’ or field with ‘Field Name’ is not found, the command is ignored. If the command has no error and the appropriate field is found, the analyzer module 29a converts a null-terminated string ‘value’ into the proper format. In the case of a Number format, the string is converted into an integer and in the case of a String field, the value is used as is.

SetField {Table Name} {Record ID} {Field Name} {Value}

For example: SetField summary cnn mod-wmt 31
If the Field ‘mod-wmt’ is number type field,
String “31” is converted into integer 31

Please amend the paragraph beginning on page 27, paragraph [0082] of the application to read as follows:

[0082] The ‘ResetField’ command is used to reset the fields of all records in a table. If a table has 20 records, and each record has a field named ‘mod-wmt,’ that field of those 20 records is reset with ‘0’. But if [Method] is set with field method such as ‘average’, ‘total’, ‘totbiggest’, the analyzer module resets only those field methods.

ResetField {Table Name} {Field Name} [Method]

For example:

Resetfield summary mod-~~wmt~~ wms

Resetfield summary onAir-~~wmt~~ wms

Resetfield summary onAir-~~wmt~~ wms total

Resetfield summary onAir-~~wmt~~ wms total+totbiggest+average

= > reset 3 property of 'onAir-~~wmt~~ wms' field.

Please amend the paragraph beginning on page 28, paragraph [0085] of the application to read as follows:

[0085] The Delete command is used to delete the table, record and/or field specified.

Delete { {Table Name} | [Record ID] | [Field Name]}

For example:

Delete Summary cnn mod-~~wmt~~ wms ➔ delete only field named 'mod-~~wmt~~ wms'

Delete Summary cnn ➔ delete whole record named 'cnn'

Delete Summary ➔ delete entire table named 'summary'

Please amend the paragraph beginning on page 30, paragraph [0091] of the application to read as follows:

[0091] FIG. 6 depicts the hierarchy from the bottom (source) tier to top (master) tier. The machine(s) executing analyzer module(s) 29a are preferably time-synched based on UTC time.

Please amend the paragraph beginning on page 30, paragraph [0093] through page 31 of the application to read as follows:

[0093] The absolute timeout time for each analyzer module Mode2 instance (Aggregating/Master Tier) is calculated based on the timetag (calculated from Interval). If the interval is 5 minutes, the current time tag received from analyzer module Mod1 is '5', and the timeout for the aggregating tier and master tier is 30 and 300 seconds, the absolute timeout for each tier is as follows:

Source	: TimeTag is 5		= 12:25:00am
Aggregatier Tie	: Timeout is 30	= Time Tag + 30 sec	= 12:25:30am
Master Tier	: Timeout is 300	= TimeTag + 300 sec	= 12:30:00am

Please amend the paragraph beginning on page 32, paragraph [0099] of the application to read as follows:

[0099] Common settings (i.e., settings used for Mode1 or Mode2) include, but are not limited to: (1) specification of mode, that is, whether the analyzer module 29 is executing in Mode1 or Mode2; (2) Listen IP and Listen Port; (3) PushIP and Push Port; and (4) Interval. Analyzer modules in Mode1 or Mode2 need to specify from which IP address it receives data. For Mode1, the analyzer module 20 uses Listen IP and Listen Port to listen for UDP packets ~~than~~ that contain analyzer commands from other programs such as a parser module 41. For Mode2, the analyzer module 20 uses Listen IP and Listen Port to bind a socket where an analyzer module in Mode1 can push data. The PushIP and Push Port pair is the destination to which an analyzer module pushes data. The Interval is the sampling rate used by an analyzer module in Mode1. The hierarchy of analyzer modules, however, need to be aware of this value to calculate the data sample time from a received time tag.

Please amend the paragraph beginning on page 34, paragraph [00104] of the application to read as follows:

[00104] With continued reference to Fig. 5, the first priority of the real-time log reporting system is to report the current connected client count and the peak connected client count for each media server. The parser module 41 uses 'Total' and 'TotalBiggest' methods for its number field definition to get the current connection count and peak connection count.

Table 12: Data Used for Marketing

CUSTOMER (ex: CNN, MTV)	# Current Clients	# Peak Clients
OnAir Real	21	64
OnStage Real	34	55
OnDemand Real	30	108
OnAir ~~WMT~~ WMS	400	554
OnStage ~~WMT~~ WMS	311	202
OnDemand ~~WMT~~ WMS	231	213

As stated above the total number of fields is the number of services multipled by the number of media types.

Please amend the paragraph beginning on page 34, paragraph [00105] through page 35 of the application to read as follows:

[00105] The parser module 41 configuration has information on how to create tables and fields. The commands required to create the table and record format shown in table 11, for example, are as follows:

<ex: Table name= "Summary", Customer = "CNN">

Register summary cnn OnAir-real num total+totbiggest+etotbiggest

Register summary cnn OnStage-real num total+totbiggest+etotbiggest

Register summary cnn OnDemand-real num total+totbiggest+etotbiggest

Register summary cnn OnAir-~~wmt~~ wms num total +totbiggest+etotbiggest

Register summary cnn OnStage-~~wmt~~ wms num total+totbiggest+etotbiggest

Register summary cnn OnDemand-~~wmt~~ wms num total+totbiggest+etotbiggest

The 'etotbiggest' method means that 'totbiggest' value must be reset at every interval, back to the 'total'. 'Total' means current number of connected clients. Whenever a new client connects, parser module 41 sends "+1"; when a client disconnects, it sends "-1". The total value means total count of currently connected clients.

Please amend the paragraph beginning on page 35 paragraph [00106] of the application to read as follows:

[00106] As explained previously, whenever a new customer (e.g. ABC, FOX, etc) appears in the log data, parser module 41 registers the related fields and if there was no table or record to house them, analyzer module 29 automatically creates it. If new data comes in, parser module 41 finds the field to be updated. The commands below show that how those commands would look like.

```
Setfield summary cnn OnAir-real 1
Setfield summary cnn OnDemand-wmt 1
Setfield summary cnn OnDemand-wmt 1
Setfield summary cnn OnDemand-wmt-1
Setfield summary cnn OnAir-real-1
Setfield summary cnn OnAir-real 1
Setfield summary cnn OnAir-real 1
```

On executing those command, the value of OnAir-real would be '2 = 1-1+1+1' and OnDemand-wmt would be '~~2~~ 1 = 1+1-1".

Please amend the paragraph beginning on page 35 paragraph [00108] through page 36 of the application to read as follows:

[00108] Once the tables are aggregated on the root tier, it connects to the Java app server 43 and sends a snapshot of the tables using XBM. When the root tier sends a snapshot of a table, it uses an XML-based table description format. A sample XML table description is shown below. An XBM call is made as many times as analyzer module 29 has records and tables. Following sample shows 2 XBM calls.

```
#call 1
<analyzer module 29-root version="1.0" date="2000-0601" time="23:00">
```

```
 <Table Name="Summary" Total="1" Current="1">
  <Record Name="MTV" Total="2" Current="1">
   <Field Type="Num" Name="OnAir-real" Total="20" TotBiggest="38"/>
   <Field Type="Num" Name="OnStage-real" Total="42" TotBiggest="532"/>
   <Field Type="Num" Name="OnDemand-real" Total="12" TotBiggest="29"/>
   <Field Type="Num" Name="OnAir-~~wmt~~ wms" Total="440" TotBiggest="332"/>
   <Field Type="Num" Name="OnStage-~~wmt~~ wms" Total="523" TotBiggest="231"/>
   <Field Type="Num" Name="OnDemand-~~wmt~~ wms" Total="124" TotBiggest="63"/>
  </Record>
 </Table>
</analyzer module 29-root>

# call 2
<analyzer module 29-root version="1.0" date="2000-0601" time="23:00" >
 <Table Name="Summary" Total="1" Current="1" >
  <Record Name="MTV" Total="2" Current="1" >
   <Field Type="Num" Name="OnAir-real" Total="67" TotBiggest="438" / >
   <Field Type="Num" Name="OnStage-real" Total="82" TotBiggest="322" / >
   <Field Type="Num" Name="OnDemand-real" Total="133" TotBiggest="29" / >
   <Field Type="Num" Name="OnAir-~~wmt~~ wms" Total="240" TotBiggest="332" / >
  <Field Type="Num" Name="OnStage-~~wmt~~ wms" Total="513" TotBiggest="131" / >
   <Field Type="Num" Name="OnDemand-~~wmt~~ wms" Total="24" TotBiggest="63" / >
  </Record>
 </Table>
</analyzer module 29-root>
```

Please amend the paragraph beginning on page 37 paragraph [00111] of the application to read as follows:

[00111] The data mining and analysis system 11 is advantageous in that, among other reasons, an application can register its own variable when it launches and send information as it registered. If the application needs to change or add a variable format or list, it can simply send an update command to the corresponding analyzer module 29a. The analyzer module 29a maintains the analyzed information and servers it to higher level analyzer modules until the root tier analyzer module summarizes the information obtained from all lower level analyzers. The data mining and analysis system 11 of the present invention abstracts mathematical and scaling aspects of different uses to provide essentially real-time reporting and to allow use with a nearly infinitely large network. The trending and dynamic ability to scale the analysis components of the system 11 has many valuable uses such as performing real-time voting. The system 11 can be configured such that the analysis of the voting results is distributed in a manner that requires a central monitoring location to poll only a few remote analyzer modules 29a. Accordingly, the system 11 provides a useful way to trend metrics in a network, as well as receive statistical data from on the order of millions of interactive end-users 22.

Please amend the paragraph beginning on page 37 paragraph [00112] of the application to read as follows:

[00112] As stated previously, any network software device, such as media server plug-in 21, can be configured to communicate with a local analyzer module ~~20~~ 29a and instruct it to start trending or analyzing new information. For voting, an edge node device can register a new variable with its parent analyzer module 29a and indicate that it wants to be analyzed, even though the analyzer modules in the system 11 were not previously configured to collect and analyze voting information. Other nodes that try to register the new variable are ignored; however, they are permitted to send data (e.g., a vote) that affects the requested analysis. In other words, an 'analysis bean' can be created and introduced to a system of analyzer modules 29a, and other nodes can participate in affecting the analysis of the 'bean'. The data mining and analysis system 11 of the present invention therefore provides a scalable way to obtain statistical information about a network (e.g., network 12), as well as introduce new metrics without having to reconfigure the analysis software.

Please amend the paragraph beginning on page 38 paragraph [00113] of the application to read as follows:

[00113] Further, by utilizing a multi-tier analyzer deployment, server information can be collated or aggregated at various points in the network, thereby reducing the stress on the network. When a query is generated, it can be answered from information stored in the local database which is populated by the remote analyzers or video server events in a real-time manner. This allows for a statistical query to be answered with very little stress on the network and a specific request to be aggregated using standard queries to the entire network. Thus, all the servers can be polled for detailed information only when needed. The stress on the network is directly proportional to the detail of the request for information. In other words, the more detailed the information that is needed, the more information that is requested from the servers. However, if the information is statistical information, this can be gathered from remote statistical software applications that are each responsible for smaller clusters of servers. One example is where a video server sends information about every request it receives. A local analyzer can keep track of the top ten requests. A parent device to that analyzer can then use these top ten requests to create a new top ten between all of its children analyzers. The top analyzer can then generate a list of the top ten requests for the entire network, while the other analyzers keep track of their respective and more localized top ten lists.